Automated Theory Formation: The Next Generation

Simon Colton[1], John Charnley[1], and Alison Pease[2]

[1] Computational Creativity Group
Department of Computing, Imperial College, London
[2] School of Electronic Engineering and Computer Science,
Queen Mary, University of London

Abstract. Automated Theory Formation was introduced as a computational model for how mathematical theories could be formed from the bare minimum such as a set of axioms or some background concepts in a domain of pure mathematics such as group theory, graph theory or number theory. The approach consists of using production rules to form new concepts from old ones, and employing a set of measures of interestingness to drive a heuristic search. Empirical pattern-based conjecture making techniques are used to find relationships between concepts, and third party automated reasoning systems are employed to prove the truth of the conjectures or find counterexamples. Automated Theory Formation has been applied to a number of discovery tasks which have led to the publication of mathematical results in the literature. In recent years, the approach has been extended, improved and generalised. This has been done via appeals to the philosophy of mathematics, namely Lakatos's suggestions in the book Proofs and Refutations describing ways in which mathematicians find and respond to counterexamples, using them to evolve concepts, conjectures and proofs within a mathematical theory. The resulting computational approach extends Automated Theory Formation by modelling ways in which mathematicians discuss counterexamples and their implications for the conjecture at hand. The recent extensions also appeal to cognitive science results, in particular Baar's suggestions in the Global Workspace Architecture theory of mammalian consciousness. The resulting computational approach involves configuring a framework for the combination of reasoning systems leading to more sophisticated and applied theory formation approaches. We survey these extensions here, in addition to other projects that have added to our understanding of Automated Theory Formation. We also frame these approaches within a broader picture, specifically with reference to Mathematical Theory Exploration approaches. We end with a discussion of future applications and extensions of Automated Theory Formation.

1 Introduction

The simulation of aspects of mathematical reasoning has been a driving force throughout the history of Artificial Intelligence research. One of the main motivations for such studies has been the desire to discover powerful, generic algorithms for applications beyond domains of mathematics. For instance, automated theorem provers are regularly employed in hardware and software verification applications and constraint solvers are used in a large range of applications areas such as scheduling and packing. Another main motivation for simulating reasoning processes has been to add to the body of mathematical knowledge, i.e., by software helping mathematicians to derive results that they otherwise would not find, and by software autonomously finding novel concepts, theorems and proofs. For instance, computer algebra systems are regularly used in mathematical research, and theorem provers, constraint solvers, SAT solvers and other systems have contributed to the discovery of novel mathematical results. A third, less obvious, motivation for simulating mathematical reasoning has been to study creative processes in people and in software. By simulating the creation of concepts, the discovery of conjectures, the derivation of proofs and the finding of counterexamples, and by simulating ways in which all of the processes can interact, it has been possible to shed light on ways in which human mathematicians create/discover new mathematics, and to suggest more generic ways in which creative software can be built.

Our contribution to automated reasoning has been to develop and nurture a novel approach called *Automated Theory Formation* (ATF), which describes a model of mathematical theory formation whereby concepts are invented to describe and categorise examples of mathematical objects such as numbers, graphs or groups; conjectures are made which relate the concepts; and proofs and counterexamples are sought to determine the truth of the conjectures. All three motivations given above have played a part in the development of ATF. That is, our approach has led to the implementation of novel, generic AI algorithms, for machine learning, constraint solving, theorem proving, computer algebra and combinations thereof. Our approach has also been used in mathematical discovery tasks, leading to new results published in the literature. Moreover, ATF is one of the approaches studied within Computational Creativity research, and has added to our understanding of creative processes in people and software.

An overview of ATF and its successes during the first stage of its development is given in section 2. However, we are more interested here in the projects from the second phase which have led to the implementation and study of more sophisticated mathematical reasoning techniques. In particular, in section 3, we describe how the philosophy of mathematics developed in the book Proofs and Refutations by Lakatos [61] was implemented in order to validate and clarify Lakatos's suggestions, and to advance the state of the art in Automated Theory Formation. The resulting computational approach extends ATF by modelling ways in which mathematicians discuss counterexamples and their implications for the conjecture at hand. In section 4, we briefly introduce an implementation of a framework for the combination of reasoning systems which is based on Baar's global workspace theory of mammalian consciousness [7, 8]. The resulting computational approach involves configuring a framework for the combination of reasoning systems leading to implementations of sophisticated mathematical reasoning systems which were only possible via ad-hoc methodologies previously.

In section 5, we put the ATF approach into context both historically, and with respect to related approaches such as in Mathematical Theory Exploration systems. In section 6, we discuss Polya's legacy and its relevance for ATF. Finally, in section 7, we discuss various future directions for the ATF approach, and we describe a new implementation which enables more fine-grained user-driven control of the theory formation process.

2 Automated Theory Formation and the HR system

By the term *Automated Theory Formation* (ATF), we mean the routine introduced in [24] and expanded upon in [33], whereby a mathematical theory including examples; concepts which categorise the examples; conjectures which relate the concepts; and proofs which demonstrate the truth of some of the conjectures; is built from some fundamental information about a mathematical domain. In algebraic domains, this fundamental information might be the axioms of the finite algebra under study, but in other domains it might differ, for instance, in number theory, a mathematical theory can be built from just two background concepts, namely addition and multiplication. In overview, the ATF routine comprises three stages:

- New concepts are generated from old ones using one of a number of production rules, as described in [29]. Concepts comprise a number of facets, most notably a formal definition in one of a number of representations (e.g., in first order logic when first order theorem provers are being used as part of the ATF routine), and a set of example tuples which satisfy the definition (which can be seen as the success set in logic programming terminology).
- Conjectures relating concepts are formed empirically by looking for patterns in the examples satisfying the concepts, as described in [26]. In particular, if no example tuples can be found for a concept, a non-existence conjecture is made stating that the concept's definition is inconsistent with the axioms of the domain. Similarly, if the tuples satisfying concept C all satisfy the definition of concept D, then an implication conjecture $C \rightarrow D$ is made, and if C and D have exactly the same examples, then the conjecture $C \leftrightarrow D$ is made.
- Third party theorem provers such as Otter [73] are employed to try and prove each conjecture, and if these fail, then model generators such as MACE [72] are used to try and find a counterexample. If a proof is found for an equivalence conjecture $C \leftrightarrow D$, then concept D is not allowed into the theory, as doing so would result in a duplication of effort. Similarly if a non-existence conjecture is proved, then the concept with the inconsistent definition is not allowed into the theory. If a counterexample is found, then this usually results in the concept being allowed into the theory, and the counterexample being used to try and disprove other open conjectures.

The ATF routine has been implemented firstly in HR1, a Prolog implementation describe in [24], and then in HR2, a Java implementation described in [33], which is still in use. Each application of the routine is an attempt to add a new concept to the theory which may result in a new concept, a conjecture (with or without a proof), a new example (introduced as a counterexample) or nothing being added to the theory. In this way, a rich theory can be built up. The theory formation is driven by a heuristic search which uses measures of interestingness to determine which old concept(s) to build new ones from at each application of the ATF routine. Such measures of interestingness are described in [32].

The value of the ATF approach has been demonstrated on numerous occasions, in the following ways:

– Through the discovery of new results published in the mathematical literature, e.g., in number theory [23] and – as part of a larger system comprising numerous automated reasoning programs – in loop and quasigroup theory [89, 90].
– By contributing to previously human-only mathematical databases, such as the Encyclopedia of Integer Sequences [31] and the TPTP library [36].
– Through the improvement, enhancement and comparison of standard AI techniques such as constraint solving [20], machine learning [30] and theorem proving [35, 99].

In addition, as described in the following sections, the ATF routine has provided inspiration for more sophisticated mathematical theory formation approaches.

3 An implementation of Lakatos-style reasoning

Taking inspiration from the writings of the philosopher of mathematics, Imre Lakatos, the HRL system is implemented in an agent architecture consisting of a number of students and a teacher, in keeping with the dialectical aspect described by Lakatos in [61]. Each agent has a copy of the HR system, and starts with a different database of examples to work with, and different interestingness measures. Students send conjectures, concepts, counterexamples, or requests such as barring a specific object of interest from the theory, to the teacher. The teacher sends requests to the students such as "work independently", "send a concept to cover counterexamples $[x, y, z]$", or "modify faulty conjecture C". The students use the methods prescribed by Lakatos to modify a faulty conjecture.

Lakatos [61] explicitly outlines six methods for modifying mathematical ideas and guiding communication: surrender, monster-barring, lemma-incorporation, exception-barring, monster-adjusting and proofs and refutations. Of these, the three main methods of theorem formation are monster-barring, exception-barring, and the method of proofs and refutations [61, p.83]. Crudely speaking, monster-barring is concerned with concept development, exception-barring with conjecture development, and the method of proofs and refutations with proof development. However, these are not independent processes; much of Lakatos's work stressed the interdependence of these three aspects of theory exploration.

In the next subsections, we provide overviews of the computational models of these processes, as implemented in HRL. In the illustrative examples that follow, we define $P \rightsquigarrow Q$ to mean that it is *nearly* true that $P \to Q$, *i.e.*, there are many supporting examples and few counterexamples. A more exact definition is given in [76], where full details of all these methods can be found.

3.1 The method of monster-barring

Monster-barring is a way of excluding an unwanted counterexample to a conjecture by (re)defining the concepts in the conjecture so as to exclude the 'counterexample'. As an example, we take Euler's conjecture that for all polyhedra, $V - E + F = 2$, where V is the number of vertices, E the number of edges and F the number of faces. For this conjecture, the hollow cube (a cube with a cube-shaped hole in it) is proposed as a counterexample, since $V - E + F = 16 - 24 + 12 = 4$. The argument is then made that the hollow cube is not a polyhedron and thus is no threat to Euler's conjecture. The definition of the concept of polyhedron then becomes the focus of the discussion. Thus, the original conjecture remains, but the meaning of the terms within it may change.

Our monster-handling algorithm Students send each other examples if they arise as counterexamples to a conjecture which the group is discussing. If a student receives an example which it has not seen before, then the student performs these steps:

1. **Decide whether to perform monster-barring.** If the new object is either a counterexample to more than a user-defined percentage of the student's total conjectures, or if it causes other objects to become counterxamples, then perform monster-barring. Otherwise, add the new object to the theory and reject the conjecture under discussion as being false.
2. **Generate a new definition of the concept.** Generate a new, intensional or extensional definition of C, which excludes the monster.

When a student receives a proposal to bar a monster, it evaluates whether it agrees or disagrees with the proposal. It does this by calculating the percentage of its own conjectures that the proposed 'monster' breaks, and voting on whether to bar the object or not, depending on a user-defined monster-barring minimum. The decision is then made democratically: if the object is to be barred, then all students downgrade the 'monster' from full example to 'pseudo object' in their theories. Pseudo objects do not count as counterexamples so cannot threaten conjectures, but are around in the theory and can be upgraded to full example at a future stage. If an explicit definition is given, then the students have to replace their old concept definition with the new one. Alternatively, if the example is to be accepted, it is added to each student's theory.

Illustrative example This demonstrates the generation of an explicit concept definition. The students debate whether the object $A1 =< P(\{a, b\}) \setminus \phi, \cup >$ is a group or not. That is, $A1$ is an algebra consisting of the set of subsets of $\{a, b\}$

except the empty set, and the operation set union. Since the empty set is not included, this object has no identity and therefore elements have no inverse.
Input: We ran the agency with three students and a teacher. The first student started with 14 examples of groups, up to size 8, and the other two students started with two algebras, $A1$, which is of size 3 and has no identity and no inverse for any of the elements, and $A2$, which is of size 4 and has an identity but no inverse element for two of the elements in it. All students start with the core concepts of group elements, the operator function, identity and inverse. We add the concept of the existence of an identity element in a group to the first student's theory. This concept would be found automatically within 40 theory formation steps and we include it early only for the purposes of this example. The monster-barring minimum is set to 15%, and the students are set to suggest an explicit definition if they perform monster-barring. The teacher asks the students to work individually for 40 steps and then submit their best conjectures.
Run Results: The first student sent the conjecture that something is a group if and only if it has an identity element. The second and third students both sent the first algebra in their theories, $A1$ as a counterexample. As this is a new object for the first student, it checks how many of its conjectures it breaks and finds that it breaks 19%. Since this is greater than the minimum set, the student proposes to bar $A1$ as a monster and sends the statement that $A1$ is not a group, as a group must have an inverse element for every element in the group. The other two students check whether $A1$ is a counterexample to any of their conjectures, find that it is not, and reject the proposal to bar it, saying that it *is* a group, as a group is any set of objects with an element in it. As the first student is out-voted, the teacher tells it to add $A1$ to its theory.

3.2 The method of exception-barring

Lakatos's treatment of exceptions is noteworthy for two reasons. Firstly, he highlights their role in mathematics — traditionally thought of as an exact subject in which the occurrence of exceptions would force a mathematician to abandon a conjecture. Secondly, Lakatos showed how exceptions, rather than simply being annoying problem cases, which we may be able to dismiss as monsters, can be used to further knowledge. *Piecemeal exclusion* is one way to deal with exceptions. It does this by excluding a type of example from the conjecture, in order to exclude a whole class of counterexamples. This is done by generalising from a counterexample to a class of counterexamples which have certain properties. For instance, the students generalise from the hollow cube to *polyhedra with cavities*, and then modify Euler's conjecture to 'for any polyhedra without cavities, $V - E + F = 2$'. Thus exceptions are seen as objects which are valid (as opposed to monsters) and force us to modify a faulty conjecture by changing the domain to which it refers. *Strategic withdrawal* is a further method which does not directly use counterexamples. Instead, it uses positive examples of a conjecture and generalises from these to a class of object, and then limits the domain of the conjecture to this class. For instance, the students generalise from the regular polyhedra to *convex polyhedra*, and then modify Euler's conjecture to 'for any convex polyhedra, $V - E + F = 2$'.

Our piecemeal exclusion algorithm Given a near equivalence conjecture $P \leftrightsquigarrow Q$, a student agent will get the list of counterexamples and determine whether each counterexample satisfies the definition of P or Q (using the set analogy, we say that a counterexample is *in* P or Q). There are three cases, given below along with what the student agent does in each case:

Case 1: all counterexamples are in P

(i) look for a concept X in the theory which exactly covers the counterexamples. If unsuccessful, and there are few counterexamples, say $[x, y, z]$, then make the concept X of being x, y or z.
(ii) form the new concept $P \wedge \neg X$
(iii) add the new concept to the theory, which will force the formation of this conjecture: $P \wedge \neg X \leftrightarrow Q$

Case 2: all counterexamples are in Q

As for case 1, but instead forming the concept $Q \wedge \neg X$ and the conjecture $P \leftrightarrow Q \wedge \neg X$

Case 3: there are counterexamples in both P and Q

(i) find a concept X in the theory which exactly covers the counterexamples in P. If unsuccessful, and there are few counterexamples, say $[x, y, z]$, then make the concept X of being x, y or z.
(ii) form the new concept $P \wedge \neg X$
(iii) form the concept $P \wedge \neg X$
(iv) add this new concept to the theory, which will force the formation of this conjecture: $P \wedge \neg X \rightarrow Q$
(v) repeat steps (i) to (iii), swapping P and Q.

If a student gets a near implication, then there can only be counterexamples on one side, *i.e.* either in concept P or in Q (but not both), and so the steps in either case 1 or case 2 are performed.

Illustrative example This demonstrates concept-barring on a near-equivalence conjecture. Using three student agents and a teacher, we get the conjecture that an integer is non-square if and only if it has an even number of divisors.
Input: Student 1 starts with the integers $1-10$ as examples and the background concepts of integer, divisor and multiplication. It is set to make implication conjectures empirically by noticing subsumptions, and to use piecemeal exclusion. Before running it independently, we manually add the concepts of square numbers and integers with an even number of divisors. Student 2 starts with the

same input as Student 1, except with integers $11 - 50$. Student 3 starts with integers $51 - 60$ and the additional manually-added concept of integers having an even number of divisors.
Run results: The teacher sends a request to work independently for 20 steps, and then send back their best implication conjecture. Student 3 forms the conjecture that all integers have an even number of divisors, and sends it to the teacher, who puts it on the group agenda for modification. The other students then find counterexamples $\{1, 4, 9\}$ and $\{16, 25, 36, 49\}$, respectively. They then generate the concept of squares and, from this, the concept of non-squares. This is used to form the conjecture that every non-square has an even number of divisors.

3.3 The method of proofs and refutations

As the title of Lakatos' book suggests, this is the most important method, to the extent that the rest of the book is often seen by commentators and critics as a lead up to this method, for instance [42]. It starts off as the method of *lemma incorporation*, and is developed via the dialectic into the method of *proofs and refutations*. Lemma incorporation works by distinguishing global and local counterexamples. The former is one which is a counterexample to the main conjecture, and the latter is a counterexample to one of the proof steps (or lemmas). A counterexample may be both global and local, or one and not the other. When faced with a counterexample, the first step is to determine which type it is. If it is both global and local, *i.e.* there is a problem both with the argument and the conclusion, then one should modify the conjecture by incorporating the guilty proof step as a condition. If it is local but not global, *i.e.* the conclusion may still be correct but the reasons for believing it are flawed, then one should modify the guilty proof step but leave the conjecture unchanged. If it is global but not local, *i.e.* there is a problem with the conclusion but no obvious flaw in the reasoning which led to the conclusion, then one should look for a hidden assumption in the proof step, then modify the proof and the conjecture by making the assumption an explicit condition. Proofs and refutations consists of using the proof steps to suggest counterexamples (by looking for objects which would violate them). For any counterexamples found, it is determined whether they are local or global counterexamples, and then lemma incorporation is performed.

Our algorithms for lemma-incorporation We interpreted Lakatos's method of lemma-incorporation as a series of algorithms which operate as follows:

1. **Determine which type of lemma-incorporation to perform.** If there are any counterexamples to the global conjecture, then if either: *(i)* these are also counterexamples to any of the lemmas in the proof, or *(ii)* there is a counterexample to a local lemma, and there is a concept which links the local counterexample to the global counterexample and this concept appears in one of the local lemmas, then perform *global and local lemma-incorporation*. If there is a global counterexample but neither *(i)* nor *(ii)* holds, then perform

hidden lemma-incorporation. Otherwise, if there are counterexamples to any of the lemmas in the proof, then perform *local-only lemma-incorporation*.

2. **Perform local-only lemma-incorporation.** Given a conjecture to which there are no known counterexamples, and a proof tree which contains a faulty lemma, $P \rightsquigarrow Q$, to which there is at least one counterexample, then if there is a concept C in the theory which exactly covers the counterexamples (or such a concept can be formed), then make the concept $P \wedge \neg C$, replace the faulty lemma with the conjecture $P \wedge \neg C \rightarrow Q$, and return the improved proof-scheme.
3. **Perform global and local lemma-incorporation.** Given a proof-scheme where there are counterexamples to the global conjecture, $P \rightsquigarrow Q$, and these counterexamples are also counterexamples to a lemma in the proof, $R \rightsquigarrow S$; form the concept C as: 'objects which satisfy the faulty lemma', by merging the two concepts R and S (this is done by using a production rule to compose R and S), modify the global conjecture by making the new conjecture $C \rightarrow Q$, and replace the old global conjecture in the proof-scheme with the modified version.
4. **Perform global-only lemma-incorporation.** Given a proof-scheme where there are counterexamples to the global conjecture, but these counterexamples are not counterexamples to any of the lemmas in the proof, and none of the lemmas have counterexamples which are related to the global counterexamples; let the global conjecture be a near-implication $P \rightsquigarrow Q$. Then go through the proof-scheme and take each lemma in turn, testing each to see whether the global counterexample is surprising because it behaves differently to supporting examples with respect to the lemma. If it is, then return this lemma as the hidden faulty lemma, and generate another conjecture, to which the global counterexample *is* a counterexample, as the explicit lemma. If not, then go through the proof-scheme and take each lemma in turn, testing each to see whether the global counterexample is meaningful with respect to the lemma. If so, then return this lemma as the hidden faulty lemma, and generate another conjecture, to which the global counterexample *is* a counterexample, as the explicit lemma. If an explicit lemma has been found, then generate an intermediate proof-scheme in which the hidden faulty lemma is replaced by the explicit lemma, perform global and local lemma incorporation on the intermediate proof-scheme, and return the result. If no lemma is surprising in either sense, then return the proof-scheme unchanged.

Working in the polyhedra domain, and given the proof tree, conjecture and counterexample as input, HRL has replicated all three of Lakatos's types of lemma-incorporation. Further details are given in [76].

4 The GC Combined Reasoning Framework

In [17, 19], we demonstrated a novel approach to combined reasoning by implementing a framework based upon the cognitive science theory of the Global

Workspace (GWT) [7, 8]. The framework considers individual reasoning tasks as separate parallel processes attached to a central workspace (figure 1). Each reasoning round begins with a broadcast on the workspace which the attached processes can ignore or react to. If they choose to react, they can propose a broadcast for the next round. Should more than one proposal be suggested, then the highest ranked is chosen, with all others being discarded. However, the processes themselves are responsible for assigning an importance rating to their proposals, meaning that the decision of which broadcast to consider next is essentially distributed.

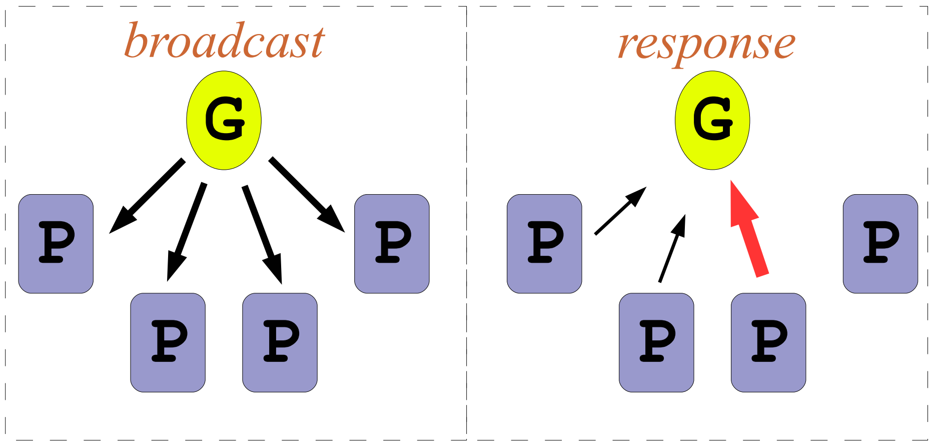

Fig. 1. The Global Workspace Architecture.

The motivation for such a framework arises from the ad-hoc nature of previous combined reasoning systems, such as HR [25] and ICARuS [21]. Although successful, these systems are often difficult to understand and inflexible to extension. The GC toolkit, which implements the GWT framework, provides a generic platform for the development of combined systems. It has been applied to a number of combined reasoning tasks, including automated theory formation and constraint problem reformulation, as described below. In each case, it has enabled an effective system to be built whilst maintaining clarity of implementation, ease of extension and the ability to easily distribute processing and relevancy checking over many hosts. Several other frameworks for combined reasoning feature in the literature. For example, the Open Mechanised Reasoning Systems (OMRS) approach [45] and the *Ω*ANTS framework [88]. However, previous projects differ from GC in various ways. In particular, GC is truly generic, with no particular application domain in mind. The framework shares common aspects with blackboard architectures [75], although the scope of the current context is reduced in GC. Similarly, the framework is akin to a multi-agent system [98] although the behaviour of the processes differs from that of agents. For instance, they lack autonomy and have the ability to detach from the workspace if so pre-directed.

4.1 Applications to Automated Theory Formation

One success of the GC toolkit was the implementation of a distributed automated theory formation system, GC-ATF. This application used the same approach to theory formation as the HR system, namely background concepts, production rules etc., as described above. Like HR, GC-ATF makes use of a number of disparate reasoning approaches, including theorem proving and model generation, as well as concept formation and conjecture making. We describe below how these tasks are encapsulated as processes and how they interact via broadcast proposals to form entire theories. We applied the resulting system to applications in algebra and, as described below, extended it to consider experiments in number theory and theorem modification. These applications provide new insight into the HR approach to automated theory formation and demonstrate the utility of GC as a development tool. We only provide brief details here, for full details please see [17].

Implementation Details In order to create a GC configuration, a developer must describe a language for the broadcast proposals, the behaviour of each of the workspace processes (including the scheme to rate the importance of proposals) and an initial state of the system. Each of these is specific to the domain of the application and once these are defined, the system operates in the GWT inspired manner described above.
Broadcasts and Proposals The broadcast language for GC-ATF consists of labelled text records, made up of a type identifier plus data specific to that particular type of broadcast. The broadcast type falls into one of five categories:

- **Background:** These broadcasts initiate theory formation. They convey information about the background concepts of the theory and their examples. In mathematical domains, they might also provide the background axioms.
- **Definitions:** These broadcasts describe new definitions for concepts that have been created through the action of production rules on existing concepts. GC-ATF represents such definitions using Prolog-style first-order predicates.
- **Concepts:** Not all definitions go on to become concepts, for example all concepts must be unique, meaning that repeat definitions are rejected. These broadcasts describe those definitions that have been retained as concepts of the theory. In addition, they carry details of the examples that satisfy the concept, distinguishing concepts with unique example sets which makes them worthy of further development.
- **Conjectures:** These broadcasts carry information of the conjectures identified by the system through the analysis of concept example sets. They might also contain a translation into theorem proving input syntax.

- **Explanations:** Proofs (or refutations) of conjectures can be found through appeals to provers and model generators. These broadcasts convey the details of those attempts, such as the proof length and the axioms used.

Workspace Processes In GC-ATF, each process performs some well-defined aspect of the overall theory formation task, falling into one of eight sub-tasks:

- **Background:** These processes construct the *Background* proposals to initialise theory formation. For instance, they may be configured to propose background concept broadcast proposals or details of the mathematical axioms.
- **Definition Creators:** Production rules are encapsulated as processes which respond to concept broadcasts by proposing new *Definition* broadcasts. They can combine multiple concepts by using a spawning mechanism whereby copies of themselves remember a specific concept for use in future combinations with other concepts. To avoid loss, failed definition proposals are handed to repeater processes which keep proposing them until successful.
- **Definition Reviewer:** This has the basic task of filtering repeat definitions.
- **Example Finder:** This process provides examples for each new and unique concept. It does this by maintaining a Prolog database of the background examples of the theory, which it queries with new concept definitions.
- **Equivalence Reviewer:** This reacts to concept broadcasts and analyses the example sets to identify those that are identical. It proposes equivalence conjectures in those cases, which reduces the search space of concepts considered by GC-ATF in a similar manner to that employed by HR.
- **Conjecturing:** Several processes compare concept example sets to suggest implication conjectures. Several others react to conjecture broadcasts by simplifying, translating or excluding them for various reasons.
- **Explainers:** These processes attempt to prove or refute conjectures by encapsulating some external reasoning system, reporting their results via *Explanation* broadcasts.
- **Reporters:** Utility processes are used to record and store selected results.

Configuration and Processing GC-ATF starts with processes for each sub-task including a definition creator process for each of the production rules the user wants to be in operation. The importance rating scheme uses a simple ordering by broadcast type, with definitions being lowest ranked followed by concepts, conjectures and then explanations. Within this, the user may choose to favour particular aspects of definitions over others by weighting their characteristics. As processing continues, new definitions are created by the definition creators from existing concepts. These are reviewed for uniqueness and equivalence and, if accepted, feed back into the concept generation process. At the same time, the example sets of new concepts are monitored by conjecturing processes, whose output is tested by provers to generate theorems.

Experiments with GC-ATF The first application of GC-ATF was to theory formation in domains of finite algebra, in particular QG quasigroups. The theories that GC-ATF produced were broadly similar to those produced by HR over the same domain. In particular, GC-ATF found all the theorems that HR found when used as part of the ICARUS system [20], meaning that it could be used as a straight replacement in that system with equivalent results. Further proof of the effectiveness of GC-ATF came from the domain of number theory, where it successfully identified that odd refactorable numbers are perfect squares, a result originally found by HR, as described in [23]. Like HR, GC-ATF can also operate outside mathematics, as demonstrated by some further investigations into general first-order domains.

Comparison with HR While the overall approach to theory formation is the same in each system (i.e. production rules, background concepts and examples etc.) the implementation details are significantly different. Both systems are coded in Java with calls to external systems in HR being replaced by encapsulated processes in GC-ATF. HR's datatable representation is replaced by a Prolog representation scheme with an engine to identify examples. The most significant difference is seen in the control system which is an ordered agenda of theory formation steps in HR. In GC-ATF, overall control comes from the interaction between processes and broadcasts via the workspace, together with the importance rating scheme. HR is a much more mature system than GC-ATF, having been the subject of several years of post-doctoral research. For instance, HR has access to 20 production rules although it commonly only uses the same 6 that have been implemented in GC-ATF. HR also has 22 measures of interestingness that are used to guide the agenda compared to just 4 used by GC-ATF. HR also has a larger number of methods for generating new conjectures from existing ones and filtering out uninteresting results. Overall, GC-ATF finds all the interesting aspects of a theory that are found by HR but this is often within a much larger body of output due to a number of factors. In particular, GC-ATF generates more concepts due to implementation differences for some production rules and how HR excludes some background concepts from theory formation. There are, however, instances where GC-ATF has produced valid concepts that HR didn't, leading to new insights into this approach to concept formation in general. The higher number of concepts leads to a large increase in conjectures in GC-ATF versus HR, which is compounded by the relative lack of conjecture filtering within GC-ATF.

Other Applications To demonstrate the extensibility of the framework, GC-ATF was enhanced to undertake some previous investigations which had featured HR. In each case, the extension process was straightforward due to the simple modular nature of the underlying framework. Firstly, the number theory experiments with the HOMER system [27] were revisited by introducing the ability to appeal to computer algebra systems for background knowledge. Secondly, a system was created to recreate the functionality of the TM system for theorem modification [34]. This demonstrated the utility of the framework as a design tool by reducing the implementation timescale from several days in the original TM system [22] to approximately four hours. GC-ATF was also applied to a novel investigation in finite algebraic structures by attempting to find particularly challenging, yet simply defined, algebraic families [18].

4.2 Parallel Implementation and Constraint Reformulation

The Global Workspace Architecture was appealing as a basis for a combined reasoning framework because of its modular and parallel nature. There is very little communication, with all information transfered via the workspace. As expected, the translation of GC from a serial to a distributed framework was straightforward, using network commands to initiate remote hosts and sockets for communication. The size of the broadcast did not significantly affect the performance of the distributed system. Specifically, for broadcasts of 90 bytes (the GC-ATF average), near linear speed-ups are possible in distributed systems using a Gigabit Ethernet network of PCs. However, the actual results depend upon the application. For instance, the GC-ATF core theory formation process benefits little from distribution as the main bottleneck is in time-limited failed proof attempts (an asynchronous model is under development to address this). In the parallel implementation, new processes are spawned on the same processing host as their parent process, leading to imbalanced load problems. Therefore, in order to extract the most efficient speed increases, users must consider the initial placement of these parent processes over the network. One successful application was the re-implementation of the ICARuS constraint reformulation system [20] as a parallel system, GC-ICARuS. To deal with the load issue, it used process mirror groups which are spawned in unison on every host but only one of which is allocated the processing task. This limits the effort to one particular host for each mirror group and avoids the need for a central system load monitor and associated notification method. Overall, the approach allowed for significant improvements over the original ICARuS problem reformulation times, as described in [17].

4.3 Summary and Future Work

GC and GC-ATF have proven highly capable in underpinning and performing the kind of theory formation tasks for which HR has previously been used. In several applications, it has shown its utility as a design toolkit by reducing the time required to create complex systems and providing clear and modular systems. GC-ATF currently lacks much of the refinement that was built into HR over many years. However, GC-ATF is currently being improved to introduce the advanced features that it lacks, whilst the general approach is being reviewed in light of insights gained from the GC-ATF implementation. Several enhancements are planned for GC, such as an improved user interface and a more faithful interpretation of the underlying cognitive theory [16]. In addition, its features allowing users to find processing capacity on the network are to be automated, so that parallel runs are initialised optimally. GC-ATF is, therefore, very well placed to become a highly effective distributed implementation of HR's approach to automated theory formation. Future projects include plans to create an automated reasoning repository, based upon GC. This would provide a toolkit for AI practitioners to harness the benefits of combining reasoning systems without the need to develop complex bespoke systems. It is hoped that this will be come a collaborative venture, with many system contributions, resulting in a rich and evolving platform for application and experimentation.

5 Exploration versus formation

5.1 A short history of automated mathematics

> …"in his subsequent design for an Analytical Engine Mr. Babbage has shown that material machinery is capable, in theory at least, of rivalling the labours of the most practised mathematicians in all branches of their science." [55, p. 498]

The field of automated mathematics today shares some similarities with its history. The first automated system, the mechanical calculator (known as the Pascaline), was an adding machine that could perform additions and subtractions directly and multiplication and divisions by repetitions, and was conceived by Pascal in 1642 while reorganizing tax revenues [38]. Subsequent early systems include Müller's universal calculating machine in 1784, which he built for the purpose of calculating and printing numerical tables: he invented this when he had to check and recalculate some tables relating to the volumes of trees [66, p. 65]. Thirty seven years later, Babbage invented his famous difference engine: an automatic, mechanical calculator designed to tabulate polynomial functions. This was inspired by a flawed table of logarithms and the idea that machines would be quicker and more reliable [11]. It is interesting to note background and motivation: while Pascal and Babbage were mathematicians, with an interest in engineering, Müller was an engineer with limited mathematical knowledge. All three systems were conceived as an aid to mathematicians, as well as scientists, accountants and surveyors.

Differences in background and motivation continue to be relevant today. While the majority of work in automating mathematics has been in symbolic manipulation and theorem proving, we are concerned here with other aspects of mathematics, including the construction of concepts, axioms, conjectures, examples and theorems. This is varyingly known as "Automated Theory Formation" [65, 25], "Automated Mathematical Theory Exploration" (the two AUTOMATHEO workshops held so far dereive their name from this phrase), "Mathematical Theory Exploration" [14] (also sometimes prefaced with "Computer-Aided" or "Algorithm-Supported"), "Automated Mathematical Discovery" [40, 32, 18], "Concept Formation in Discovery Systems" [48], and "Automated Theorem Discovery" [71]. Such a plethora of terminology can be unhelpful and can mask similarities between the different fields. In particular, the twin strands of Automated Theory Formation and Automated Mathematical Theory exploration seem to be developing independently without a clear differentiating methodology. Below we discuss commonalities and differences between the two schools of thought.

5.2 Automated Theory Formation

Automated theory formation, the focus of this paper, derives its terminology from psychology in which the term “concept formation” is used (see, for example, [13]) to describe the search for features which differentiate exemplars from non-exemplars of various categories. Lenat used this term in his 1977 paper: *Automated Theory Formation in Mathematics* [65] and we have extended it to describe a family of techniques used to construct and evaluate concepts, conjectures, examples and proofs (embodied in the HR system and derivatives, described above).

When Lenat built the AM system [65], there were systems which could define new concepts for investigation, such as [Winston 70], and systems which could discover relationships among known concepts, such as Meta-Dendral [Buchanan 75]. No system could perform both of these tasks: Lenat saw this as the next step. AM was designed to both construct new concepts and conjecture relationships between them; fully automating the cycle of discovery in mathematics. Lenat describes this as follows:

> What we are describing is a computer program which defines new concepts, investigates them, notices regularities in the data about them, and conjectures relationships between them. This new information is used by the program to evaluate the newly-defined concepts, concentrate upon the most interesting ones, and iterate the entire process. [ref]

AM is a rule-based system which uses a framelike scheme to represents its knowledge, enlarges its knowledge base via its collection of heuristic rules, and controls the firing of these rules via its agenda mechanism. Lenat chose elementary arithmetic as the development domain because he could use personal introspection for the heuristics for constructing and evaluating concepts. Given the age of this discipline, Lenat thought it unlikely that AM would make significant discoveries, although he did cite its “ultimate achievements” as the concepts and conjectures it discovered (or could have discovered). He suggested various criteria by which his system could be evaluated, many of which focused on an exploration of the techniques. For instance, he considered generality (running AM in new domains) and how finely-tuned various aspects of the program are (the agenda, the interaction of the heuristics, etc). Lenat saw his system and future developments in this field as having implications for mathematics itself (finding results of significance), for automating mathematics research (developing AI techniques), and for designing “scientist assistant” programs (aids for mathematicians). This shows a broad spread of motivation. Despite the seeming success of the AM system, it is one of the most criticised pieces of AI research, for instance see chapter 13 of [24] for an argument of why many of the claims made by Lenat about AM were false.

5.3 Mathematical Theory Exploration

The phrase Mathematical Theory Exploration seems to have originated with Buchberger [give earliest ref]. His motivation is to support mathematicians during their exploration of mathematical theories. This support is intended to be for the straightforward reasoning, which he argues, covers most mathematical thought, rather than then ingenious points, which he leaves for human mathematicians. Buchberger's long term goal is provide routine tools for the exploration activity of working mathematicians, to support the invention and structured build-up of mathematical knowledge. His Theorema project aims at prototyping features of a system for such a purpose. Its features include Integration of the Functionality of Current Mathematical Systems (retention of the full power of current numerics and computer algebra systems, as well as enabling the user to add their own algorithms to the system); Attractive Syntax (input and output is readable and presented attractively, and can be personalised by the user); and Structured Mathematical Knowledge Bases (tools are provided for building and using large mathematical knowledge libraries). Buchberger has evaluated the potential of this strategy by illustrating the automated synthesis of his own Gröbner bases algorithm.

5.4 Commonalities and differences between MTE and ATF

We believe that ATF and MTE share many goals and would benefit from a closer alignment: both fields are highly specialised and by joining forces (while being explicit about differences) the techniques being developed would have greater impact. The main commonalities between MTE and ATF are that both are interested in the same aspects of mathematical thinking, aiming to automatically construct and evaluate elements of a mathematical theory, including concepts, conjectures, theorems, axioms and examples. Both strands contrast themselves with Automated Theorem Proving. The main differences between these two approaches seem to be that in MTE:

- the main proponents define themselves as mathematicians (they hold a PhD and and have experience in research mathematics);
- the primary motivation is to support mathematicians;
- systems are user-interactive;
- systems are specific to mathematics,

while in ATF:

- the main proponents define themselves as AI researchers (they often have a mathematics background up to Masters level, but their PhD and research experience is in AI);

– the primary motivation is to extend AI techniques; a second motivation is to produce interesting new mathematics;
– systems are fully automated;
– systems can be applied to non-mathematical domains.

The different backgrounds of the people who named the fields “automated theory formation” (Lenat) and “mathematical theory exploration” (Buchberger) perhaps reflects different metaphysical perspectives on invention (in which case new mathematical ideas are being *formed*, created or produced) and discovery (in which case abstract mathematical objects exist independently of us and are *explored* or investigated) in mathematics. There is a vast literature on the Platonic and the constructivist view of mathematics, but we shall not venture down this path here (interested readers are referred to [52, 87]). Instead, for now, we make the pragmatic assumption that while there may well be different cognitive processes involved in invention than those involved in discovery, the fields of ATF and MTE are not currently concerned with this level of detail and so the philosophical distinction is not relevant for our purposes.

6 Polya’s legacy and its relevance for ATF

6.1 The informal face of mathematics

George Polya was a mathematician and an educator, with an interest in characterising ways in which people reason and problem-solve in mathematics. He published his insights in four influential books [80–83], which have been of particular interest to the educational community, and have spawned a new body of work in the philosophy of mathematics. This new direction focuses on mathematical practice, that is, how mathematicians do mathematics. Polya argued, and showed by example, that it is pertinent to discuss current or historical practice of mathematics from a philosophical perspective, and that much can be said about the non-rigourous, non-Euclidean face of mathematics-in-the-making. This face of the coin compliments the rigourous, Euclidean face of mathematics, which has traditionally been the main point of focus for philosophers and educators. This history is paralleled in our attempts to automate mathematical thinking: researchers in ATF and MTE are interested in informal aspects of mathematics and believe that we can automate at least some such aspects, while the mainstream focus has been on the rigourous face of mathematics via automated theorem provers and proof checkers.

Lenat cites Polya as an inspiration for his work and in doing so shows us an important but sometimes overlooked point; that developers of systems in ATF and MTE can find inspiration in work by historians of mathematics [60], pyschologists [63], eduators [62] and mathematicians themselves [49] (all cited in Lenat’s [64]).

One of Polya’s main contributions was to identify some heuristics involved in the invention of mathematics. For instance, he suggested problem-solving strategies such as “look at simple cases and try to find a pattern”, “explore a specific case" and "explore the conditions of the problem". He also recommended using analogy and scientific induction to help to solve a problem ("Can you find a problem analogous to your problem and solve that?" "Can you solve your problem by deriving a generalization from some examples?"), two forms of reasoning which he went on to emphasize and explore in [81]. He defined analogy as structure-preserving mappings, giving as examples analogies between addition and multiplication of numbers (both are commutative, associative and admit an inverse relation); subtraction and division; and zero and one. This work has inspired computational accounts, such as emulations of his work on reasoning by analogy [41, 58], how to solve it by computer [39] and an important account on how to select between different problem-solving methods [43] (difficulties in selecting between Polya's heuristics have come to be seen as a weakness in his theory by some educators).

6.2 The "Maverick" tradition'

Polya was a major influence on Lakatos, a researcher in mathematics who translated Polya's *How To Solve It* into Hungarian, and subsequently became an eminent philosopher of mathematics and science. Lakatos developed Polya's research programme of a philosophical analysis of informal mathematics, focusing on social aspects of mathematics and showing how outliers are used in mathematics to help to determine the scope of a problem, guide concept formation, and improve faulty proofs. Lakatos, together with Kitcher [57], Aspray [5] and Tymoczko [95], firmly set the agenda for a breakaway "maverick" tradition in the philosophy of mathematics, in which mathematical practice takes centre stage (as opposed to the traditional focus on epistemological and ontological issues). This small, but rapidly growing, body of work is interdisciplinary to varying degrees, but always bridges mathematics, the history of mathematics and the philosophy of mathematics. Recent work includes [15, 37, 47, 54, 56, 70, 96]. Also highly relevant to this field are introspective accounts from mathematicians, including well-known work by Poincaré [79, pp. 1- 16], Hadamard [49], Wilder [97] and Hersh [51], as well as many memoirs, including Hardy's famous apology [50], and two volumes of interviews with contemporary mathematicians [3, 4]. This body of work is of immense value to the ATF/MTE community.

The mavericks investigate questions such as how mathematical knowledge grows; what counts as mathematical progress; what makes some mathematical ideas (or theories) better than others; what constitutes a mathematical explanation; how mathematical discovery might take place and what the relationship between discovery and justification in mathematics might be; and how new discoveries are structured and integrated into existing knowledge (see, for example, the introductory sections of [5, 70, 47] for these and other questions). These are very similar questions to those asked, or at least answered implicitly via computational systems in ATF and MTE. For instance, Colton [25] provided one way in which mathematical knowledge can grow via his production rules in the HR system. He also suggested and implemented a pragmatic response to the question about what makes some mathematical ideas better than others, via his interestingness measures [32].

McCasland [71] is also investigating interestingness in mathematics via his classification in MATHsAiD of proved statements as facts (results of no intrinsic mathematical interest), lemmas (statements which might be useful in the proof of subsequent theorems), or theorems (either routine or significant results). Aberdein has studied relationships between informal arguments and formal arguments in mathematics, and has related his ideas to the argumentation community, which has a computational focus [1, 2]. Leading on from this, we have developed our implementation of Lakatos's ideas (discussed above in section 3), in which informal mathematical proof was represented using work from the argumentation community [77]. Similarly, we have addressed the relationship between discovery and justification by implementing Lakatos's methods in which conjectures are discovered to help to improve a proof and a proof improved to help to discover new conjectures.

6.3 Aspects of mathematical thinking

There is not space here to go into detail on the work which has arisen from the new maverick direction. Here, we merely outline several areas which may be of interest to the ATF/MTE communities and point to relevant work.

1. **Analogies.** Sinaceur [10] shows how different types of process can be used to discover formal analogies, and also shows how formalisation and axiomatisation contribute to the art of invention. Breger [12] also discusses ways in which finding analogies between a given problem and an extended class of related problems can help towards generalisation. Knobloch [59] uses historical case studies to identify different ways in which analogy contributes to mathematics: how it can help between a specific and the general case, how it can work within a proof, how it can help to formulate problems, and its role in the development of concepts. Knobloch calls analogies "the least misleading" instrument of discovery we have. Schlimm [86] argues that the standard structure-mapping account of analogies does not apply to mathematics and instead proposes an account of analogies based on axioms.
2. **Axiomatisation.** Sinaceur [10] suggests how axiomatisation can be used in the discovery process, for example allowing mathematicians to see previously unsuspected relationships, as well as showing the reasons for the relationship. Schlimm [85] has also written extensively on the role of axiomatics in discovery in mathematics, showing how it drives the invention of new theories and notions.
3. **Heuristics for discovery.** Polya's [80] is the seminal work for heuristics in mathematical discovery. Other works include Muntersbjorn's [74], in which she argues that the development of more general problem solving strategies is an important aspect of mathematical progress, and Hersh's [53], in which he analysed a mathematical case study (his own proof of Heron's area formula, which was much simpler than the classical proof) to help him to

formulate general observations about how new mathematical results might be discovered.

4. **Progress.** There have been several attempts at a taxonomy of mathematical progress, for instance Kitcher's account of mathematical progress [5], in which he distinguishes different tpes of progress; Peckhaus [78] and Grosholz's [46] idea that old theories are not replaced by new ones, rather exist side by side, and Thiel's identification of important factors of progress in a mathematical discipline [93]. Criteria which an important result should satisfy are often considered to be general: however, Maddy [69] considers criteria which vary in different areas of mathematics and in different historical episodes, and argues that progress occurs locally. This provides some justification for systems which have different methods of evaluation in different domains.

Philosophers of mathematical practice are investigating many other topics which are highly relevant to researchers in ATF and MTE. For instance, Avigard [6] discusses the role of computers in aiding mathematical discovery, and examines what he considers to be a symbiotic relationship between accounts of mathematical understanding and Artificial Intelligence. Furthermore, Tappenden [91, 92] discusses whether there are any objective features which fruitful mathematical concepts share; Breger [12] considers the relationship between examples and generalisations or abstractions; Barabashev [9] investigates the relationship between mathematics and empirical reality in the growth of mathematical knowledge and Gillies [44] argues that some simple mathematical concepts are embodied in the world.

Just as we, as researchers in ATF and MTE, can trace back our intellectual lineage to Polya, we should be looking to our cousins in the philosophy of mathematical practice to help guide current and future directions of our work.

7 Future Directions

In addition to the two major projects described above, there are a number of smaller projects with Automated Theory Formation which have been carried out, and a number of more substantial projects which are work in progress. In the PhD research of Pedro Torres, the question of generalising ATF at a meta-level has been looked at [94]. By starting with a valuable, user-given theory, T, and a seed theory, S, from which T was derived, the system being built in this project is able to analyse the concepts in T and derive the first order production rules which enable the construction of T from S. The hypothesis being tested is whether the theories produced by further usage of these production rules (i.e., to produce theories additional to T) will be of more value than could be generated by a generic production rule approach such as that implemented in the HR system.

In the PhD research of Ramin Ramezani, the question of whether ATF can be applied to realistic investigation problems is being looked at [84]. Such investigation problems are posed as dynamic constraint satisfaction problems which change at various time steps, and for which some of the constraints are hidden in data from historical case studies. We argue that such problems more realistically capture the state of affairs in problems such as medical diagnosis or criminal investigations, where a culprit has to be found, than standard AI problem formulations such as straight constraint problems or machine learning problems. ATF is being used here to mine the constraints held in the case studies as part of the solution finding process. We aim to show that – while neither a straight theorem proving process, constraint solving process nor machine learning process can solve these problems – a combination of reasoning techniques including ATF can efficiently identify culprits in such investigation problems.

With respect to showing that ATF has value in non-mathematical domains, the work of Llano et al. has shown that the approach has much potential for usage in software verification projects, [67, 68]. In particular, they find that ATF can be useful in finding invariants in software specifications written in the Event-B modeling language. In addition, we have shown that theory formation can be useful in an artistic context, in particular, by inventing fitness functions for scene constructions by generative art software [28]. We plan to take this work further via the usage of more sophisticated fitness function generation techniques for the generation of three dimensional scenes.

Finally, in a development which we hope will rejuvinate the usage of the ATF approach in mathematical discovery tasks, we have taken inspiration from the MTE projects described above, and we are currently building a more interactive, real-time generative mathematics system. The system is named HR-Taxi[3] and implements an *always running*, multi-core approach to theory formation that can be driven by the user. That is, in one thread the system is constantly forming a theory in an exhautive way using the ATF routine as discussed above. The user is presented with the top level concepts to choose from to start their exploration of the domain. When they click on one, because the first rounds of theory formation have already been undertaken, the user is instantly presented with all the new concepts which can be built from their choice. In a separate window, the user is presented with all the conjectures that are empirically true about their most recent choice of concept.

As the exploration continues, the user will eventually choose a concept which hasn't been fully expanded, and this spawns a new theory formation process in a separate thread which starts from their choice and forms a theory specifically about that concept and how it relates to the others already defined. In an age of multi-core processors in every computer we use, such a parallel approach seems a sensible way to use the additional processing power available. In early tests, we have been able to use the system to quickly re-discover certain conjectures

[3] HR is named after mathematicians Hardy and Ramanujan, hence the *Taxi* affectation seemed appropriate, given the allusion to the user driving the process and the well known anecdote involving the number of the taxi that Hardy took to see Ramanujan in hospital.

found originally by HR, for instance that odd refactorable[4] numbers are perfect squares. We aim to show that ATF can be used as an effective MTE approach which can help mathematicians to find interesting new concepts and conjectures in domains of interest to them.

8 Conclusion

We have described here Automated Theory Formation as an active area of research, by looking at both recent and "historical" progress in this field. In particular, we have looked in detail at two projects which have extended and improved upon the original theory formation routines in the HR system by implementing more sophisticated mathematical reasoning techniques via appeal to philosophical and psychological theories. We have further described ongoing and future work in ATF which we hope will lead to this approach being used as a generic reasoning tool in the AI practitioner's toolbox, and specifically for user-driven mathematical discovery tasks.

We have also argued that ATF shares many important features with the research undertaken in the Mathematical Theory Exploration community. We have contextualised this by describing early (and ongoing) systems in both ATF and MTE, and identified commonalities and differences. We believe that greater congruency between the two fields would be useful to both. We have also argued that, given the debt ATF owes to Polya, and the existence of a subsequent Polya-style body of work, both fields would greatly benefit from a reading of this work. We have described some example topics within this body of work, including analogical thinking, axiomatisation, heuristics for discovery, ways in which progress is measured in mathematics, the role of computers in aiding mathematical discovery, fruitful mathematical concepts, and the relationship between examples and generalisations or abstractions. We believe that if MTE and ATF realign themselves more closely and take note of the new work in the philosophy of mathematics, then this may carry us closer towards realising our goal of formalising aspects of informal mathematics.

References

1. A. Aberdein. The uses of argument in mathematics. *Argumentation*, 19:287–301, 2005.
2. A. Aberdein. Managing informal mathematical knowledge: Techniques from informal logic. In J. M. Borwein and W. M. Farmer, editors, *MKM 2006, LNAI 4108*, pages 208–221, Berlin, 2006. Springer-Verlag.
3. D. Albers, G.L. Alexanderson, and C. (Eds) Reid. *More Mathematical People: Contemporary Conversations.* Harcourt Brace and Jovanovich, 1989.
4. D. Albers and G.L. (Eds) Alexanderson. *Mathematical People: Profiles and Interviews.* Birkhäuser, Boston, 1985.

[4] A refactorable number n is such that the number of divisors (including n) itself divides n.

5. W. Aspray and P. (Eds.) Kitcher. *History and philosophy of modern mathematics.* University of Minnesota Press, Minneapolis, USA, 1988.
6. J. Avigard. Computers in mathematical inquiry. In P. Mancosu, editor, *[70]*, pages 302–316. Oxford University Press, USA, 2008.
7. B Baars. *A cognitive theory of consciousness.* Cambridge University Press, 1988.
8. B Baars. *In the theater of consciousness: The workspace of the mind.* Oxford University Press, 1997.
9. A. Barabashev. Evolution of the modes of sytemtization of mathematical knowledge. In P. Mancosu, editor, *[70]*, pages 315–330. Oxford University Press, USA, 2008.
10. H. Benis-Sinaceur. The nature of progress in mathematics: the significance of analogy. In E. Grosholz and H. Breger, editors, *[47]*, pages 281–294. Springer, 2000.
11. B. V. Bowden (Ed). *Faster Than Thought: A Symposium on Digital Computing Machines.* Pitman Publishing, London, UK, 1953.
12. H. Breger. Tacit knowledge and mathematical progress. In E. Grosholz and H. Breger, editors, *[47]*, pages 221–230. Springer, 2000.
13. J. Bruner, J. J. Goodnow, and G. A. Austin. *A study of thinking.* Science Editions, New York, 1967.
14. B. Buchberger. Mathematical theory exploration. *Invited talk at IJCAR. www.easychair.org/FLoC-06/buchberger_ijcar_floc06.pdf*, 2006.
15. C. Cellucci and D. Gillies. *Mathematical reasoning and heuristics.* Kings College Publications, London, 2005.
16. J Charnley. Applying the GC combined reasoning framework to mathematical discovery. In *Proceedings of the AISB Symposium on Mathematical Practice and Cognition*, 2010.
17. J Charnley. *A Global Workspace Framework for Combined Reasoning.* PhD thesis, Imperial College, London, 2010.
18. J Charnley and S Colton. Applications of a global workspace framework to mathematical discovery. In *Proceedings of the Conferences on Intelligent Computer Mathematics workshop on Empirically Successful Automated Reasoning for Mathematics*, 2008.
19. J Charnley and S Colton. A global workspace framework for combining reasoning systems. In *Proceedings of the Symposium on the Integration of Symbolic Computation and Mechanised Reasoning (Calculemus)*, 2008.
20. J Charnley, S Colton, and I Miguel. Automatic generation of implied constraints. In *Proceedings of the 17th European Conference on AI*, 2006.
21. J Charnley, S Colton, and I Miguel. Automatic generation of implied constraints. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, 2006.
22. S Colton. Personal communication.
23. S Colton. Refactorable numbers - a machine invention. *Journal of Integer Sequences*, 2, 1999.
24. S Colton. *Automated Theory Formation in Pure Mathematics.* Springer-Verlag, 2002.
25. S. Colton. *Automated Theory Formation in Pure Mathematics.* Springer-Verlag, 2002.
26. S Colton. The HR program for theorem generation. In *Proceedings of the Eighteenth Conference on Automated Deduction*, 2002.
27. S Colton. Automated conjecture making in number theory using HR, Otter and Maple. *Journal of Symbolic Computation*, 39(5):593–615, 2005.

28. S Colton. Automatic invention of fitness functions, with application to scene generation. In *Proceedings of the EvoMusArt Workshop*, 2008.
29. S Colton, A Bundy, and T Walsh. HR: Automatic concept formation in pure mathematics. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 1999.
30. S Colton, A Bundy, and T Walsh. Automatic identification of mathematical concepts. In *Machine Learning: Proceedings of the 17th International Conference*, 2000.
31. S Colton, A Bundy, and T Walsh. Automatic invention of integer sequences. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 2000.
32. S. Colton, A. Bundy, and T. Walsh. On the notion of interestingness in automated mathematical discovery. *International Journal of Human Computer Studies*, 53(3):351–375, 2000.
33. S Colton and S Muggleton. Mathematical applications of Inductive Logic Programming. *Machine Learning*, 64:25–64, 2006.
34. S Colton and A Pease. The TM system for repairing non-theorems. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR) workshop on Disproving*, 2004.
35. S Colton and A Pease. The TM system for repairing non-theorems. In *Selected papers from the IJCAR'04 disproving workshop, Electronic Notes in Theoretical Computer Science*, volume 125(3). Elsevier, 2005.
36. S Colton and G Sutcliffe. Automatic generation of benchmark problems for automated theorem proving systems. In *Proceedings of the Seventh AI and Maths Symposium*, 2002.
37. D. Corfield. *Towards a philosophy of real mathematics.* Cambridge, Cambridge University Press, 2003.
38. M. d'Ocagne. Le calcul simplifié. *Annales du Conservatoire national des arts et métier. 2e série*, 5,, 1893.
39. R. G. Dromey. *How to Solve It by Computer.* Prentice Hall, 1982.
40. S. L. Epstein and N. S. Sridharan. Knowledge representation for mathematical discovery: Three experiments in graph theory. *Journal of Applied Intelligence*, 1(1):7–33, 1991.
41. T. G. Evans. A program for the solution of geomtric-analogy intelligence test questions. In M. Minsky, editor, *Semantic Information Processing*, pages 271–353. The MIT Press, Cambridge, Massachusetts, 1968.
42. S. Feferman. The logic of mathematical discovery vs. the logical structure of mathematics. In P. D. Asquith and I. Hacking, editors, *Proceedings of the 1978 Biennial Meeting of the Philosophy of Science Association*, volume 2, pages 309–327. Philosophy of Science Association, East Lansing, Michigan, 1978.
43. E. Fink. How to solve it automatically: Selection among problem-solving methods. In *Proceedings of AIPS*. AAAI, 1998.
44. D. Gillies. An empiricist philsophy of mathematics and its implications for the history of mathematics. In P. Mancosu, editor, *[70]*, pages 41–58. Oxford University Press, USA, 2008.
45. F Giunchiglia, P Pecchiari, and C Talcott. Reasoning theories: Towards an architecture for open mechanized reasoning systems. *Journal of Automated Reasoning*, 26:291–331, 2001.
46. E. Grosholz. The partial unification of domains, hybrids, and the growth of mathematical knowledge. In E. Grosholz and H. Breger, editors, *[47]*, pages 81–92. Springer, 2000.

47. E. Grosholz and H. (Eds.) Breger. *The Growth of Mathematical Knowledge (Synthese Library)*. Springer, 2000.
48. K. Haase. Discovery systems. Technical Report 898, MIT, 1986.
49. J. Hadamard. *The Psychology of Invention in the Mathematical Field*. Dover, 1949.
50. G. H. Hardy. *A Mathematician's Apology*. Cambridge University Press, 1940.
51. R. Hersh. Fresh breezes in the philosophy of mathematics. *The American Mathematical Monthly*, 102(7):589–594, 1995 (Aug. - Sep.).
52. R. Hersh. *What is mathematics, really?* Oxford University Press, UK, 1997.
53. R. Hersh. On the interdisciplinary study of mathematical practice, with a real live case study. In Jean Paul van Kerkhove, Bart van; Bendegem, editor, *[?]*, pages 231–241. Springer, 2007.
54. R. (Ed.) Hersh. *18 Unconventional Essays on the Nature of Mathematics*. Springer, 2005.
55. W. S. Jevons. On the mechanical performance of logical inference. *Philosophical Transactions of the Royal Society of London*, 160:497–518, 1870.
56. Jean Paul van (Eds.) Kerkhove, Bart van; Bendegem. *Perspectives on Mathematical Practices: Bringing Together Philosophy of Mathematics, Sociology of Mathematics, and Mathematics Education*. Springer. Logic, Epistemology, and the Unity of Science, Vol. 5, 2007.
57. P. Kitcher. *The Nature of Mathematical Knowledge*. Oxford University Press, Oxford, UK, 1983.
58. R. E. Kling. Reasoning by analogy with applications to heuristic problem solving: A case study. Technical report, Stanford Artificial Intelligence Project Memo AIM-147, CS Department report CS-216, August, 1971.
59. E. Knobloch. Analogy and the growth of mathematical knowledge. In E. Grosholz and H. Breger, editors, *[47]*, pages 294–314. Springer, 2000.
60. E. Koppelman. Progress in mathematics. In *Workshop on the Historical Development of Modern Mathematics*, July, 1975.
61. I. Lakatos. *Proofs and Refutations*. CUP, Cambridge, UK, 1976.
62. W. E. Lamon. *Learning and the Nature of Mathematics*. Science Research Associates, Palo Alto, 1972.
63. G. R. Lefrancols. *Psychological Theories and Human Learning*. Wadsworth Publishing, Belmont, Ca., 1972.
64. D. Lenat. *AM: An Artificial Intelligence approach to discovery in mathematics*. PhD thesis, Stanford University, 1976.
65. D. B. Lenat. Automated theory formation in mathematics. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 833–842, Cambridge, MA, 1977. Morgan Kaufmann.
66. C. G. Lindgren, M. (translated by McKay. *Glory and Failure: The Difference Engines of Johann Müller, Charles Babbage, and Georg and Edvard Sheut*. The MIT Press, USA, 1990.
67. T Llano, A Ireland, and A Pease. Discovery of invariants through automated theory formation. In *In Proceedings of the 15th International Refinement Workshop. EPTCS 55*, 2011.
68. T Llano, A Ireland, A Pease, S Colton, and J Charnley. Using automated theory formation to discover invariants of event-b models. In *Rodin User and Developer Workshop*, 2010.
69. P. Maddy. Mathematical progress. In E. Grosholz and H. Breger, editors, *[47]*, pages 341–352. Springer, 2000.
70. P. (Ed) Mancosu. *The Philosophy of Mathematical Practice*. Oxford University Press, 2008, USA.

71. R. McCasland and A. Bundy. MATHsAiD: a mathematical theorem discovery tool. In *Proceedings of SYNASC, to appear*, 2006.
72. W McCune. A Davis-Putnam program and its application to finite first-order model search. Technical Report ANL/MCS-TM-194, Argonne National Laboratories, 1994.
73. W.W. McCune. Otter 3.0 Reference Manual and Guide. Technical Report ANL-94/6, Argonne National Laboratory, Argonne, USA, 1994.
74. M. Muntersbjorn. The quadrature of parabolic segments 1635 - 1638: A response to herbert breger. In E. Grosholz and H. Breger, editors, *[47]*, pages 231–256. Springer, 2000.
75. A Newell. Some problems of basic organization in problem-solving systems. In *Proceedings of the second conference on self-organizing systems*, pages 393–342, 1962.
76. A. Pease. *A Computational Model of Lakatos-style Reasoning.* PhD thesis, School of Informatics, University of Edinburgh, 2007. Online at http://hdl.handle.net/1842/2113.
77. A. Pease, A. Smaill, S. Colton, and J. Lee. Bridging the gap between argumentation theory and the philosophy of mathematics. *Foundations of Science*, 14(1-2):111–135, 2009.
78. V. Peckhaus. Scientific progress and changes in hierarchies of scientific disciplines. In E. Grosholz and H. Breger, editors, *[47]*, pages 363–376. Springer, 2000.
79. J.H. Poincaré. *Science and method.* Dover Publications, NY, 1952.
80. G. Polya. *How to solve it.* Princeton University Press, 1945.
81. G. Polya. *Mathematics and plausible reasoning: Induction and analogy in mathematics*, volume I. Princeton University Press, 1954.
82. G. Polya. *Mathematics and plausible reasoning: Patterns of Plausible Inference*, volume II. Princeton University Press, 1954.
83. G. Polya. *Mathematical Discovery.* John Wiley and Sons, New York, 1962.
84. R Ramezani and S Colton. Automatic generation of dynamic investigation problems. In *In Proceedings of the Automated Reasoning Workshop*, 2010.
85. Dirk Schlimm. Axiomatics and progress in the light of 20th century philosophy of science and mathematics. In B. Loewe, V. Peckhaus, and T. Rasch, editors, *Foundations of the Formal Sciences IV*, Studies in Logic, pages 233–253. College Publications, London, 2006.
86. Dirk Schlimm. Two ways of analogy: Extending the study of analogies to mathematical domains. *Philosophy of Science*, 75:178–200, April 2008.
87. S. Shapiro. *Thinking about Mathematics: The philosophy of mathematics.* OUP, Oxford, 2000.
88. V Sorge. *A Blackboard Architecture for the Integration of Reasoning Techniques into Proof Planning.* PhD thesis, Universität des Saarlandes, Saarbucken, 2001.
89. V Sorge, S Colton, R McCasland, and A Meier. Classification results in quasigroup and loop theory via a combination of automated reasoning tools. *Comment.Math.Univ.Carolin*, 49(2):319–339, 2008.
90. V Sorge, A Meier, R McCasland, and S Colton. Automatic construction and verification of isotopy invariants. *Journal of Automated Reasoning*, 40(2-3):221–243, 2008.
91. J. Tappenden. Mathematical concepts and definitions. In P. Mancosu, editor, *The Philosophy of Mathematical Practice*, pages 256–275. OUP, Oxford, 2008.
92. J. Tappenden. Mathematical concepts: Fruitfulness and naturalness. In P. Mancosu, editor, *The Philosophy of Mathematical Practice*, pages 276–301. OUP, Oxford, 2008.

93. C. Thiel. On some determinants of mathematical progress. In E. Grosholz and H. Breger, editors, *[47]*, pages 407–415. Springer, 2000.
94. P Torres and S Colton. Automated meta-theory induction in pure mathematics. In *Proceedings of the Automated Reasoning Workshop*, 2008.
95. T. Tymoczko, editor. *New directions in the philosophy of mathematics.* Princeton University Press, Princeton, New Jersey, 1998.
96. B. Van Kerkhove, J.-P. van Bendegem, and J. (Eds.) De Vuyst. *Philosophical Perspectives on Mathematical Practice.* College Publications, 2010.
97. Wilder. *Evolution of Mathematical Concepts.* John Wiley and Sons, Inc, NY, 1968.
98. M Wooldridge. *Introduction to MultiAgent Systems.* John Wiley and Sons Inc, 2002.
99. J Zimmer, A Franke, S Colton, and G Sutcliffe. Integrating HR and tptp2X into MathWeb to compare automated theorem provers. In *Proceedings of the CADE'02 Workshop on Problems and Problem sets*, 2002.